Strategy Synthesis for Mean-Payoff Expression Objectives

Yaron Velner

The Blavatnik School of Computer Science, Tel Aviv University, Israel

Abstract. Mean-payoff expressions are the closure of mean-payoff objectives under the algebraic operations of min, max and sum. This class of objectives was introduced in [1], and the decidability of the verification problem (that is, one-player games) for such objective was established. In this paper, we study, for the first time, synthesis problems for mean-payoff expression objectives, and the most relevant problem is the synthesis of a finite-memory controller. This problem is captured by two-player mean-payoff expression games on graph, where the objective of one player is to maximize the value of the expression, the objective of the other player is to minimize the value, and player 1 is restricted to finite-memory strategies. Our main contribution is an effective algorithm that computes the optimal value that player 1 can achieve by a finite-memory strategy. (More accurately, the algorithm computes the least upper bound on the achievable values.)

1 Introduction

The algorithmic theory of infinite games on graphs is a powerful and flexible framework for the design of reactive systems. In games played on graphs, a pebble is placed on an initial vertex, and in every round, a player moves the pebble to a successor vertex. The vertices are partitioned into player-1 and player-2 vertices, and in every round, the player who owns the vertex in which the pebble is located, advance the pebble into an adjacent vertex. The process of choosing next vertex for the pebble is repeated forever, and the outcome of such infinite process is a *play*.

Traditionally, in the framework of verification and synthesis, games on graphs have been studied with *qualitative* objectives such as reachability, safety and ω-regular conditions. Recently, there is an emerging interest in games with *quantitative* objectives; in these games, player 1 wishes to maximize the value of the play, and the objective of player 2 is to minimize that value.

The classical quantitative objective is the *mean-payoff* objective. In this objective, every transition in the graph has a weight, and the mean-payoff of a play is the long-run average weight of the play until round n, as n goes to infinity. With mean-payoff objectives we can express quantitative properties such as average response time of an arbiter, or average power consumption of a machine. However, the mean-payoff objective is not *robust*, as it is not closed under the algebraic operations of $\max, \min$ and sum [1]. For this reason, Chaterjee et al. [1] introduced the class of mean-payoff expressions objectives, which is the closure of mean-payoff objectives under the operations min, max and sum. They proved that the verification problem is decidable for mean-payoff expression specifications, and as a result, the class of mean-payoff expressions is the first (and currently the only) known natural class of quantitative objectives that is both robust and has an effective algorithm for the verification problem. However, they did not investigate the corresponding synthesis problem, and no synthesis algorithms were known.

In this work, we study for the first time the synthesis problem for mean-payoff expression objectives. The synthesis task for quantitative objectives is to find a player-1 strategy that is *optimal* with respect to the objective. That is, to find a strategy such that the minimal value of all plays according to this strategy is maximal. Since we are usually interested in synthesis of hardware mechanisms, the most relevant synthesis task is to find an optimal finite-memory player-1 strategy. However, for mean-payoff expression objectives, an optimal finite-memory strategy may not exists, even for a one-player game. For this reason, we concentrate on the problem of finding the least upper bound for the optimal value that player 1 can achieve by a finite-memory strategy. Our main contribution is an effective algorithm that computes this value. We note that once this value is computed, one can compute for any $\epsilon > 0$ an ϵ-optimal strategy by enumerating all player-1 finite-memory strategies, and solve the verification problem for the one-player game that is induced by this strategy (note that this process is guaranteed to halt).

For reasons of convenience, in this paper, we switch the roles of the players, and assume that the objective of player 1 is to minimize the value of the plays (by a finite memory strategy). This can be done w.l.o.g since mean-payoff expressions are closed under numerical complement, that is, if E is an expression, then there exists an expression that is equivalent to $-E$.

In the most abstract level, we solve a two-player game by finding a finite-memory strategy that induces the *worse* one-player (player-2) game graph in terms of threshold for which player 2 can win (in the one-player game). To be more concrete, we obtain a solution in four steps (described in Sections 3-6). In the first step, we give a finite description for the sets of cycles in one-player games that are induces by player-1 finite-memory strategies. In the second step, we recall that in a one-player mean-payoff expression game, the maximal value of the play depends only in the set of cycles in the graph, and we characterize the *satisfactory* sets of cycles for every threshold. In the third step, we combine the results of the first two steps, and solve games in which player 1 objective is to have a finite-memory strategy such that the cycles in the induced one-player game graph violates the mean-payoff expression (with respect to a given threshold). In the fourth and last step, we use the results of step three, and obtain our main result.

2 Preliminaries

Multidimensional weighted graphs. A multidimensional weighted graph is a tuple $G = (S, T, w : T \to \mathbb{Q}^k)$, where S is the set of vertices, T is the set of transitions, and w is a multidimensional weight function. The weight vector of a finite path $\pi = t_1 t_2 \dots t_n$ is denoted by $w(\pi) = \sum_{i=1}^{n} w(\pi)$, and we denote by $w_i(\pi)$ the projection of the weight vector to dimension i. We denote $Avg(\pi) = \frac{w(\pi)}{|\pi|}$, and $Avg_i(\pi)$ is the projection of $Avg(\pi)$ to dimension i. For an infinite path $\pi = t_1 t_2 \dots$ we denote $LimInfAvg_i(\pi) = \liminf_{n\to\infty} \frac{\sum_{j=1}^{n} w_i(t_j)}{n}$ and $LimSupAvg_i(\pi) = \limsup_{n\to\infty} \frac{\sum_{j=1}^{n} w_i(t_j)}{n}$.

Games on graph. A game graph is a directed graph $G = (S = S_1 \cup S_2, s_0, T, w)$, where S is the set of vertices; S_i are *player i* vertices; s_0 is the *initial vertex*; $T \subseteq S \times S$ is the set of edges; and $w : T \to \mathbb{Q}^k$ is a multidimensional weight function.

A *play* is an infinite path in the graph that begins in s_0. The $2k$-dimensional mean-payoff vector of a play $\pi = s_0 t_1 s_1 t_2 \dots s_n t_n \dots$ is defined as

$$MP(\pi) = (LimInfAvg_1(\pi), \dots, LimInfAvg_k(\pi), LimSupAvg_1(\pi), \dots, LimSupAvg_k(\pi))$$

Strategies A *strategy* is a recipe for the extension of a finite path. A *player-i strategy* is a function $\sigma : S^* S_i \to S$, such that for every finite path π that ends in vertex s we have $(s, \sigma(\pi)) \in T$. A strategy has *finite memory* if it can be implemented by a Moore machine $(M, m_0, \alpha_n, \alpha_u)$, where M is a finite set of memory states, m_0 is the initial memory state, $\alpha_u : M \times S \to M$ is the update function, and $\alpha_n : M \times S_i \to S$ is the next vertex function. If a play prefix is in state s_i and memory state M, then the strategy choice for the next vertex is $s = \alpha_n(M, s_i)$ and the memory is updated to $\alpha_u(M, s_i)$.

We denote the set of all player-1 finite memory strategies by $\mathcal{FM}$.

Game graph according to a finite-memory strategy For a game graph $G = (S = S_1 \cup S_2, T, w)$ and a player-1 finite-memory strategy $\sigma = (M, m_0, \alpha_u, \alpha_n)$, we denote the *game graph according to strategy* σ by G^σ, and we define it as following:

– The vertices of G^σ are the Cartesian product $S \times M$; player-i vertices are $S_i \times M$; and the initial vertex of G^σ is (s_0, m_0).
– For a player-1 vertex (s, m), the only successor vertex is $(\alpha_n(s, m), \alpha_u(s, m))$. For a player-2 vertex (s, m) the set of successor vertices is $\{(q, n) \mid (s, q) \in T \text{ and } \alpha_u(s, m) = n\}$.

Note that the out-degree of all player-1 vertices is one, and thus G^σ is a *one-player game graph.*

Mean-payoff expression objectives Given a k-dimensional game graph, a *mean-payoff expression* is defined by the following grammar rule:

$$E = LimInfAvg_1 \mid \cdots \mid LimInfAvg_k \mid LimSupAvg_1 \mid \cdots \mid LimSupAvg_k \mid \max(E, E) \mid \min(E, E) \mid E{+}E$$

The value of a play π according to an expression E is denote by $E(\pi)$ it is defined as

- $E(\pi) = \mathit{LimInfAvg}_i(\pi)$ (resp. $E(\pi) = \mathit{LimSupAvg}_i(\pi)$) if $E = \mathit{LimInfAvg}_i$ ($E = \mathit{LimSupAvg}_i$).
- $E(\pi) = \mathrm{op}(E_1(\pi), E_2(\pi))$ if $E = \mathrm{op}(E_1, E_2)$, for op $\in \{\max, \min, +\}$.

Values of strategies and games We note that a tuple (σ, τ) of player 1 and player strategies (respectively) uniquely defines a play $\pi_{\sigma,\tau}$ in a given graph. For a game graph G, mean-payoff expression E and a tuple of strategies (σ, τ) we denote $Val_{\sigma,\tau} = E(\pi_{\sigma,\tau})$. Recall that $\mathcal{FM}$ is set of all player-1 finite-memory strategies and we denote (in this paragraph only) by B the set of all player-2 strategies. The *worst case value* of a player-1 strategy τ is denoted by $Val_\sigma^{worst} = \inf_{\tau \in B} Val_{\sigma,\tau}$. The *best case value* of a player-1 strategy τ is denoted by $Val_\sigma^{best} = \sup_{\tau \in B} Val_{\sigma,\tau}$.

The *maximal value of a game* is defined as $\max_{G,E} = \sup_{\sigma \in \mathcal{FM}} Val_\sigma^{worst}$, and the *minimal value of a game* is defined as $\min_{G,E} = \inf_{\sigma \in \mathcal{FM}} Val_\sigma^{best}$. Intuitively, the maximal (resp. minimal) value of a game is the maximal (minimal) value that player 1 can ensure by a finite-memory strategy. We note that since the class of mean-payoff expressions is closed under numerical complement [1], there is a reduction from the computation of the maximal values to the computation of minimal values (and vice-versa), due to the fact that $\max_{G,E} = \min_{G,-E}$.

Quantitative and qualitative analysis For a given game graph, expression, and a rational threshold $r \in \mathbb{Q}$: The *quantitative analysis* task is to compute the maximal value of the game. The *qualitative analysis* task is to decide whether there is a player-1 finite-memory strategy σ for which $Val_\sigma^{worst} \geq r$. That is, whether player 1 can assure that a value of at least r for the expression.

Qualitative games and winning strategies. A *qualitative game* is a game on graph equipped with a winning condition $W \subseteq T^\omega$. A play $\pi \in T^\omega$ is *winning* for player 1 if $\pi \in W$, and a strategy σ is a player-1 *winning strategy* if for every player-2 strategy τ we have $\pi_{\sigma,\tau} \in W$.

3 The Linear Span Multidimensional Mean-Payoff Games

For two vectors $u = (u_1, \ldots, u_k), v = (v_1, \ldots, v_k) \in \mathbb{R}^k$, we say that $v \leq u$ if $v_i \leq u_i$ for every $i \in \{1, \ldots, k\}$. For a set $S \subset \mathbb{R}^k$, we define $DC(S)$ to be the downward closure of S. Formally, $DC(S) = \{v \in \mathbb{R}^k \mid \exists u \in S \text{ s.t } v \leq u\}$. For a finite set of vectors $V = \{v_1, v_2, \ldots, v_n\} \in \mathbb{R}^k$, we denote $SPAN(V) = \{\sum_{i=1}^n \alpha_i v_i \mid \sum_{i=1}^n \alpha_i = 1 \text{ and } \alpha_1, \ldots, \alpha_n \geq 0\}$, and we denote the downward closure of $SPAN(V)$ by $DCSPAN(V)$.

A *linear span multidimensional mean-payoff game* (in short, *LSMMP* game) is played on a k-dimensional weighted graph, and every infinite play ρ has a k-dimensional lim-inf mean-payoff vector $\underline{MP}(\rho)$. The winning objective for player 1 is given by a set of vectors $V \subseteq \mathbb{R}^k$, and player 1 wins in play ρ if $\underline{MP}(\rho) \in DCSPAN(V)$. Hence, a *LSMMP* is a tuple (G, s_0, V), where G is a multidimensional weighted graph, s_0 is the initial state, and V is a set of vectors.

The motivation for us to investigate *LSMMP* games is due to the next lemma.

Lemma 1 *Let (G, s_0, V) be a LSMMP game, then player 1 has a finite-memory winning strategy if and only if there exists a player-1 finite-memory strategy σ such that the average weight of any reachable simple cycle in G^σ is in $DCSPAN(V)$.*

Proof. In order to prove the direction from left to right, let us assume that player 1 has a finite-memory winning strategy σ for the *LSMMP* game (G, s_0, V). Towards contradiction, let us assume that in G^σ there is a reachable simple cycle C with $Avg(C) \notin DCSPAN(V)$. Let π_0 be a path to a vertex in the cycle C, then the path $\pi_0(C)^\omega$ is an infinite path that is consistent with σ and $\underline{MP}(\pi_0(C)^\omega) = Avg(C) \notin DCSPAN(V)$, which contradicts the assumption that σ is a winning strategy for the *LSMMP* game.

The intuition for the proof of the converse direction is simple. For a given graph G, we can decomposed any path to a prefix with length at most $|G|$, a set of simple cycles, and a suffix of length at most $|G|$; and the lim-inf mean-payoff of an infinite path is a convex combination of the simple cycles it visits. Hence, if for a finite strategy σ, we get that all the weights of the reachable cycles in G^σ are in $DCSPAN(V)$, then for every path consistent with σ we have $\underline{MP}(\pi) \in DCSPAN(V)$.

To formally prove the above intuition, we assume, towards a contradiction, that there is a path π in G^σ for which $\underline{MP}(\pi) = u \notin DCSPAN(V)$, and we denote by δ the (Euclidean) distance between u and $DCSPAN(V)$. By definition, for every $\epsilon > 0$, there exist infinitely many prefixes of π, such that the distance between the average weight vector of any such prefix and u is at most ϵ. In particular, for $\epsilon = \frac{\delta}{8\sqrt{k}}$, we get that there exists a finite prefix π_0, with length at least $\frac{2|G|W}{\epsilon}$, and the distance between $Avg(\pi_0)$ and u is at most $\frac{\delta}{2}$. We decompose the path π_0 to a prefix ρ_1 a set of simple cycles $C_1, \dots, C_n$ and a suffix ρ_2 such that (i) the length of ρ_1 and ρ_2 is at most $|G|$; and (ii) the cycle C_i occurs m_i times in the path π_0. We get that $Avg(\pi_0) = \frac{w(\rho_1)+w(\rho_2)+\sum_{i=1}^n m_i w(C_i)}{|\pi_0|}$; and since the weights of ρ_1 are ρ_2 at most $|G|W$ in every dimension[1], we get that $Avg(\pi_0) \leq \frac{2|G|W}{\pi_0} + \frac{\sum_{i=1}^n m_i w(C_i)}{\pi_0} \leq \frac{(2|G|W, 2|G|W, \dots, 2|G|W)}{\pi_0} + \frac{\sum_{i=1}^n m_i Avg(C_i)}{\sum_{i=1}^n m_i}$. Since $|\pi_0| \geq \frac{2|G|W}{\epsilon}$, we get that the distance between $Avg(\pi_0)$ and $\frac{\sum_{i=1}^n m_i Avg(C_i)}{\sum_{i=1}^n m_i}$ (which is in $DCSPAN(V)$) is at most $\sqrt{k}\epsilon \leq \frac{\delta}{8}$. Finally, we get that the distance between u and $Avg(\pi_0)$ is at most $\epsilon < \frac{\delta}{2}$ and the distance between $Avg(\pi_0)$ and $DCSPAN(V)$ is less than $\frac{\delta}{2}$, which contradicts the assumption that the distance between u and $DCSPAN(V)$ is δ. $\square$

Given a game graph, we say that a set of vectors V is *feasible* if player 1 has a finite-memory winning strategy for the corresponding *LSMMP* game. In this section, we wish to characterize, for a given game graph, the feasible sets of vectors.

The next lemma suggests that it is enough to characterize the sets of vectors that are feasible when player 2 is restricted to a memoryless strategy.

[1] where W is the maximal weight in the graph, in absolute value

Lemma 2 *For any LSMMP game (G, s_0, V), player 1 has a finite-memory winning strategy in the game if and only if against every player-2 memoryless strategy τ, player 1 has a finite-memory winning strategy in the (one-player) LSMMP(G^τ, s_0, V).*

Proof. The direction from left to right is trivial. Our proof for the converse direction is inspired by [2], and the key intuition of the proof, is the following. Let s be a player-2 vertex, with two out-edges t_1 and t_2. Suppose that player 2 cannot win by using only one of the edges. Then player 1 can combine the two winning strategies of the games on $G - \{t_1\}$ and $G - \{t_2\}$, and he can obtain a finite-memory strategy σ, such that the simple cycles in G^σ are composition of the cycles from the two winning strategies, and hence, σ is a winning strategy for G. (We note that we cannot use directly the result of [2] for two reasons: (i) player 1 is restricted to finite-memory strategies; and (ii) the *LSMMP* game winning objective is not a member of the class of the so called *convex winning objectives.*)

In order to formally prove the key intuition we claim that player 2 has a winning strategy in the *LSMMP* game only if it has a memoryless winning strategy, and we prove the claim by induction on the number of player-2 vertices with out-degree greater then one. The base case, where all of player-2 vertices have out-degree one, is trivial. For the inductive step, let us assume that there is a player-2 vertex s with out-edges t_1 and t_2 (if there is no such vertex, then we are in the base case). For $i = 1, 2$, let G_i be $G - \{t_i\}$. If player-2 has a winning strategy in either (G_1, s_0, V) or (G_2, s_0, V), then by the induction hypothesis, he has a memoryless winning strategy, and surely this is also a memoryless winning strategy for (G, s_0, V), and the claim follows.

Otherwise, we complete the proof by claiming that player 2 does not have a winning strategy in (G, s_0, V). Indeed, for $i = 1, 2$, let σ_i be a finite-memory player-1 winning strategy in (G_i, s_0, V), If in σ_1 (resp., σ_2), the vertex s is unreachable then it is surely a winning strategy also for (G, s_0, V). Otherwise, σ_1 induces a finite-memory winning strategy also for the game (G, s, V) (that is, a game that starts in s and not in s_0), and we denote this strategy by σ_1^s. we construct a winning strategy σ in the following way. The memory structure of σ is a tuple $(M_1, M_2, \{1, 2\})$ where intuitively M_1 is the memory structure of σ_1^s, M_2 is the memory structure of σ_2, and the third value in the tuple indicates if we are playing according to σ_1^s or σ_2. At the beginning of a play, σ decides according to σ_2 (and updates M_2 accordingly). If σ decides according to σ_2 and edge t_1 is visited, then σ decides according to σ_1^s (and updates M_1 accordingly), until edge t_2 is visited, and then σ again decides according to σ_2, and so on. We note that σ is a finite-memory strategy, and that any simple cycle in G^σ is a composition of simple cycles from $G_1^{\sigma_1^s}$ and $G_2^{\sigma_2}$.

We now show that σ is a winning strategy. By Lemma 1, it is enough to show that the weights of all the simple cycles in G^σ are in $DCSPAN(V)$. The latter holds, since every cycle in G^σ is a composition of simple cycles from $G_1^{\sigma_1^s}$ and $G_2^{\sigma_2}$, and since σ_1^s and σ_2 are winning strategies (in G_1 and G_2 respectively), then the weights of their cycles are in $DCSPAN(V)$; and thus, the average weight of any cycle that is a composition of those cycles is also in $DCSPAN(V)$.

To conclude, we get that player 2 is the winner if and only if it has a memoryless winning strategy, and the proof of Lemma 2 follows. $\square$

In a one-player game graph, we say that a vector $v \in \mathbb{R}^k$ is *achievable*, if there exists a player-1 finite-memory strategy that assures lim-inf mean-payoff vector v. We note that in a one-player game, player 1 can win $DCSPAN(V)$ if and only if there is an achievable vector $v \in DCSPAN(V)$. Hence, if for a given game graph G, we have that τ_1 and τ_2 are the only possible player-2 memoryless strategies; and for τ_1 the achievable vectors are $\{u_1, u_2\}$; and for τ_2 the achievable vectors are $\{v_1, v_2\}$; then by Lemma 2, we get that the only four possible feasible sets in an *LSMMP* game over G are $\{u_i, v_j\}$ for $i, j = 1, 2$. In the general case, there may be only finitely many player-2 memoryless strategies $\tau_1, \ldots, \tau_m$; however, against strategy τ_i, the set of achievable vectors V_i might be infinite. The next lemma describes the feasible sets, in terms of achievable vectors, for the general case.

Lemma 3 *Given game graph (G, s_0), let $\tau_1, \ldots, \tau_n$ be the only possible player-2 memoryless strategies in G, and let V_i be the achievable vectors for (G^{τ_i}, s_0); Then a set of vectors U is feasible (in (G, s_0)) if and only if there exists k vectors $v_1 \in V_1, \ldots, v_k \in V_n$ such that $DCSPAN(U) \supseteq DCSPAN(\{v_1, \ldots, v_n\})$.*

Proof. We first prove the direction from right to left. For this purpose we assume that there exists $v_i \in V_i$ for every $i \in \{1, \ldots, n\}$; and we claim that player 1 is the winner of the *LSMMP* game $(G, s_0, \{v_1, \ldots, v_n\})$. Indeed, by Lemma 2, it is enough to show that player 1 is the winner against any player-2 memoryless strategy τ_i. By definition of v_i, we get that against player-2 strategy τ_i, player 1 can achieve lim-inf mean-payoff v_i (and clearly $v_i \in DCSPAN(\{v_1, \ldots, v_n\})$), and thus player 1 is the winner, and the set $\{v_1, \ldots, v_n\}$ is feasible. Since $DCSPAN(U) \supseteq DCSPAN(\{v_1, \ldots, v_n\})$, it trivially follows that U is feasible as well, and we completed the proof for the direction from right to left.

For the converse direction, let us assume that U is feasible, and let σ be player-1 strategy that wins (G, s_0, U). Let r_i be the (only) lim-inf mean-payoff vector for the (one) play that is consistent with both σ and τ_i. By definition, $r_i \in V_i$, and $r_i \in DCSPAN(U)$, and therefore $DCSPAN(\{r_1, \ldots, r_n\}) \subseteq DCSPAN(U)$, and by choosing $v_i = r_i$ we complete the proof. $\square$

In light of Lemma 3 and Lemma 2, it is enough to obtain a characterization of achievable vectors in a one-player game. For this purpose, we present the next definitions.

When a graph is given, we say that a set of cycles $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ is a *set of connected cycles* if there exists a path in G such that its edges are exactly the edges that occur in $C_1, \ldots, C_n$. Given an initial vertex, we say that $\mathcal{C}$ is a *reachable set* if all the cycles in $\mathcal{C}$ are reachable from the initial vertex. A set of cycles is a *reachable connected simple cycles* if it is reachable, connected and contains only simple cycles. We denote the set of all sets of simple connected and reachable cycles by $SCR(G, s_0)$. For a set of cycles $\mathcal{C} = \{C_1, C_2, \ldots, C_n\} \in SCR(G, s_0)$, we define the *relative interior rational span* of $\mathcal{C}$ as

$$relRatSPAN(\mathcal{C}) = \{\sum_{i=1}^{n} \alpha_i Avg(C_i) \mid \sum_{i=1}^{n} \alpha_i = 1 \text{ and } \alpha_1, \ldots, \alpha_n \in \mathbb{Q}^{++}\}$$

and we similarly define

$$relSPAN(\mathcal{C}) = \{\sum_{i=1}^{n} \alpha_i Avg(C_i) \mid \sum_{i=1}^{n} \alpha_i = 1 \text{ and } \alpha_1, \ldots, \alpha_n \in \mathbb{R}^{++}\}$$

The next lemma characterize the set of achievable vectors in a one-player game.

Lemma 4 *For a graph (G, s_0), the set of achievable vectors is*

$$\bigcup_{\mathcal{C} \in SCR(G,s_0)} relRatSPAN(\mathcal{C})$$

Proof. The proof is straight forward. Every finite-memory strategy (in a one-player game graph) is an ultimately periodic infinite path $\pi = \pi_0(\pi_1)^\omega$, such that π_1 is a cyclic path and the mean-payoff vector of π is $Avg(\pi_1)$. A cyclic path is a composition of connected simple cycles $C_1, \ldots, C_n$, such that cycle C_i occurs m_i times in the path, and the average weight of the cyclic path is $\frac{\sum_{i=1}^n m_i w(C_i)}{\sum_{i=1}^n m_i |C_i|}$; and for $\alpha_i = \frac{|C_i| m_i}{\sum_{j=1}^n m_j |C_j|}$, we get that average weight is

$$\sum_{i=1}^{n} \alpha_i Avg(C_i)$$

We note that by definition, α_i is a positive rational, and $\sum_{i=1}^n \alpha_i = 1$. Hence the mean-payoff vector of any ultimately periodic path is in $\bigcup_{\mathcal{C} \in SCR(G,s_0)} relRatSPAN(\mathcal{C})$.

Conversely, for any vector $v \in \bigcup_{\mathcal{C} \in SCR(G,s_0)} relRatSPAN(\mathcal{C})$, we have that for some positive rationals $\alpha_1, \ldots, \alpha_n$, $v = \sum_{i=1}^n \alpha_i Avg(C_i)$. Let $q_1, \ldots, q_n$ and p be natural numbers for which $\alpha_i = \frac{q_i}{p}$, and we denote $\Pi = |C_1| \cdot |C_2| \cdot \cdots \cdot |C_n|$. Then by choosing $m_i = \frac{p \Pi \alpha_i}{|C_i|}$, we get that

$$\frac{\sum_{i=1}^n m_i w(C_i)}{\sum_{i=1}^n m_i |C_i|} = \frac{p\Pi \sum_{i=1}^n \alpha_i Avg(C_i)}{p\Pi \sum_{i=1}^n \alpha_i}$$

and since $\sum_{i=1}^n \alpha_i$, we get that

$$\frac{\sum_{i=1}^n m_i w(C_i)}{\sum_{i=1}^n m_i |C_i|} = \sum_{i=1}^{n} \alpha_i Avg(C_i)$$

Hence, for every vector in $relRatSPAN(\mathcal{C})$ there exists an ultimately periodic path with a corresponding weight vector. $\square$

By combining Lemmas 2,3 and 4 we immediately get the next proposition

Proposition 1 *Given a game graph (G, s_0), a set of vectors U is feasible if and only if there exists a set of vectors $V = \{v_1, \ldots, v_n\}$ such that (i) $DCSPAN(U) \supseteq DCSPAN(V)$; and (ii) $\tau_1, \ldots, \tau_n$ are the only possible player-2 memoryless strategies and for every vector v_i there exists a reachable connected simple cycles set $\mathcal{C}_i$ in (G^{τ_i}, s_0) such that $v_i \in relRatSPAN(\mathcal{C}_i)$.*

4 Properties of One-Player Mean-Payoff Expressions Games

In this section, we present key properties of one-player mean-payoff expressions games (with arbitrary strategies). We note that the results in this section are derived from [1, 3] in a straight forward manner.

For an expression E and a game graph (G, s_0), we say that a threshold $\nu \in \mathbb{R}$ is *feasible* if there exists a strategy (that is, an infinite path) that achieves a value of at least ν for the expression E. We will show how to characterize the range of feasible thresholds, and for that purpose we bring the next definitions and lemmas.

We say that an expression is *max-free* if the max operator does not occur in the expression. The next lemma implies that we may consider only expressions with the form of $E = \max(E_1, \ldots, E_n)$, where E_i is a max-free expression, for $i = 1, \ldots, n$.

Lemma 5 *For any game graph and expression E, there is an expression $F = (F_1, \ldots, F_n)$ such that (i) F is equivalent to E; (ii) every F_i is a max-free expression; and (iii) F is computable from E (and the game graph).*

Proof. The proof is by induction on the number of operators in the expression, which we denote by m. If $m = 0$, then E is a max-free expression and $F = \max(E, E)$ is the desirable expression. If $m > 1$, then there are two expressions E_1 and E_2, both with at most $m - 1$ operators, such that $E = \text{op}(E_1, E_2)$, for some op $\in \{\max, \min, \text{sum}\}$. By the induction hypothesis there exists $F_1 = \max(H_1, \ldots, H_n)$ (resp., $F_2 = \max(G_1, \ldots, G_n)$) that is a maximum of max-free expressions, and E_1 (E_2) is equivalent to F_1 (F_2); and we get that $F = \max(H_1, \ldots, H_n, G_1, \ldots, G_n)$ is the desirable expression. □

We recall the *max-free* constraints, which we presented in [3], and which describe the feasible thresholds for a max-free expression.

Definition 1 (Max-free constraints) *Let G be a strongly-connected k-dimensional game graph, and we denote by $\mathbb{C}$ the set of simple cycles of G. Let E be a max-free expression such that the first j dimensions of G occur in E as lim-inf (and the others as lim-sup). We define a variable X^i_c for every simple cycle c and index $i \in \{j + 1, \ldots, k\}$, and we define a vector of variables $\overline{r} = (r_1, \ldots, r_{2k})$. The the max-free constraints for threshold $\nu \in \mathbb{Q}$ are*

$$\sum_{c\in\mathbb{C}} X^i_c Avg_m(c) \geq r_m \text{ for every } i \in \{j + 1, \ldots, k\} \text{ and } m \in \{1, \ldots, j, i\} \quad (1)$$

$$\sum_{c \in \mathbb{C}} X_c^i = 1 \textit{ for every } i \in \{j+1, \ldots, k\} \tag{2}$$

$$X_c^i \geq 0 \textit{ for every } i \in \{j+1, \ldots, k\} \textit{ and } c \in \mathbb{C} \tag{3}$$

$$M_E \times \overline{r} \geq (0, \ldots, 0, \nu)^T \tag{4}$$

where M_E is a matrix that is independent of the graph, and computable from E. It was proved in [3] that a threshold ν is feasible if and only if the corresponding max-free constraints are feasible. The max-free constraints shed light on a key property of mean-payoff expressions. Let us denote for a strongly connected graph G, the set of vectors that contains the average weight vectors of all the simple cycles in the graph, by $Avg(G)$; and let us denote $DCSPAN(G) = DCSPAN(Avg(G))$. Then:

Lemma 6 *Let E be a mean-payoff expression over k dimensions, and let G be a strongly connected k-dimensional graph. If the maximal feasible threshold of E in graph G is ν, then for every strongly connected k-dimensional graph H with $DCSPAN(H) \subseteq DCSPAN(G)$ we have that the maximal feasible threshold of E in graph H is at most ν.*

Proof. Since we assume that $E = \max(E_1, \ldots, E_n)$, where E_i is a max-free expression, it is enough to prove that if the threshold ν is feasible in H for the max-free expression E_i, then it is also feasible in G. Let $C_1^G, \ldots, C_n^G$ and $C_1^H, \ldots, C_m^H$ be the simple cycles of G and H respectively. We note that since $DCSPAN(H) \subseteq DCSPAN(G)$, then for every convex combination $x_1, \ldots, x_m$, there is a convex combination $y_1, \ldots, y_n$ such that $\sum_{i=1}^m x_i Avg(C_i^H) \leq \sum_{i=1}^n y_i Avg(C_i^G)$ (in every dimension). Hence, a solution for the max-free constraints over graph H induces a solution for the max-free constraints over G (by replacing, in the inequalities of constraints 1 over graph H, the convex combination of cycles of H $x_1, \ldots, x_m$ with the convex combination of cycles of G $y_1, \ldots, y_n$).

Thus, every threshold that is feasible for H is also feasible for G, and the proof follows. □

Lemma 6 gives rise to the notion of *unsatisfactory weights vectors*. For a given expression and threshold, a set of vectors $V = \{v_1, \ldots, v_n\}$ is *unsatisfactory* if for every strongly connected graph G with simple cycles $C_1, \ldots, C_n$ such that $Avg(C_i) = v_i$, and the threshold is infeasible in G. We note that by Lemma 6, a set of vectors $V = \{v_1, \ldots, v_n\}$ is unsatisfactory if **there exists** a strongly connected graph G with simple cycles $C_1, \ldots, C_n$ such that $Avg(C_i) = v_i$, we have that the threshold is infeasible in G.

Similarly, when a multidimensional weighted graph G is given, we say that a set of simple cycles (of G) is *unsatisfactory* if the corresponding set of average weights is unsatisfactory.

We will investigate games with winning objectives that are related to unsatisfactory cycles in the next section.

5 The Unsatisfactory Cycles Game

The *unsatisfactory cycles game* (in short, *UCG* game) is played over a game graph G, a mean-payoff expression E and a threshold $\nu \in \mathbb{Q}$, and player-1 is the winner of the game (from an initial vertex s_0) if it has a finite-memory strategy σ for which the set of reachable simple cycles in G^σ are unsatisfactory (with respect to E and ν).

We note that this game is not equivalent to the synthesis problem for mean-payoff expressions, as we do not require the cycles to be in one SCC. However, it is straight forward to see that a finite-memory winning strategy in the *UCG* game is a wining strategy for the mean-payoff expression game.

In this section, we shall solve *UCG* games, that is, we will prove that there is an effective algorithm to compute the greatest lower bound on ν for which player 1 is the winner in a *UCG* game (when G and E are given). Our solution relies on properties of the *LSMMP* games (which we defined in section 3) and on properties of one-player mean-payoff expression games that we established in section 4.

The key step of the solution is due to the next lemma.

Lemma 7 *Player 1 is the winner of the UCG game* (G, s_0, E, ν) *if and only if there exists a set of unsatisfactory (with respect to* ν*) vectors* $V = \{v_1, \dots, v_n\}$ *(where* n *is the number of player-2 memoryless strategies) such that*

> *For every player-2 memoryless strategy* τ_i*, there exists a set of reachable connected simple cycles* $\mathcal{C}^{\tau_i}$ *in* G^{τ_i}*, such that* $v_i \in \mathit{relRatSPAN}(\mathcal{C}^{\tau_i})$.

Proof. We first prove the direction from left to right. Let σ be a player-1 finite-memory winning strategy for the *UCG* game (G, s_0, E, ν), and let $\mathcal{C} = C_1, \dots, C_m$ be the set of reachable simple cycles in G^σ. By Lemma 1, player 1 is the winner for the *LSMMP* game over $\{\mathit{Avg}(C_1), \dots, \mathit{Avg}(C_n)\}$, and by Proposition 1 there exists a set of vectors $V = \{v_1, \dots, v_n\}$ such that $\mathit{DCSPAN}(\mathcal{C}) \supseteq \mathit{DCSPAN}(V)$, and for every player-2 memoryless strategy τ_i, there exists a set of reachable connected simple cycles $\mathcal{C}^{\tau_i}$ in G^{τ_i}, such that $v_i \in \mathit{relRatSPAN}(\mathcal{C}^{\tau_i})$. To complete the proof for the direction from left to right, we need to show that V is unsatisfactory. This follows from the fact that $\{\mathit{Avg}(C_1), \dots, \mathit{Avg}(C_n)\}$ in unsatisfactory (by definition), and since $\mathit{DCSPAN}(\mathcal{C}) \supseteq \mathit{DCSPAN}(V)$, then by Lemma 6 we get that V is also unsatisfactory.

Conversely, if there exists a set V as stated, then by Proposition 1 there exists a player-1 finite-memory winning strategy σ for the *LSMMP* game (G, s_0, V). Let $\mathcal{C} = C_1, \dots, C_m$ be the simple cycles in G^σ. By definition $\mathit{DCSPAN}(\mathcal{C}) \subseteq \mathit{DCSPAN}(V)$, and thus, by Lemma 6, if V is unsatisfactory, then so is $\mathcal{C}$. Thus, σ is a winning strategy also for the *UCG* game (G, s_0, E, ν), and the proof follows. □

We note that for a given graph, the number of player-2 memoryless strategies is finite, and if we fix a player-2 memoryless strategy, then the number of sets of reachable connected simple cycles is also finite. For a player-2 memoryless strategy τ, let us denote by $\mathbb{C}^\tau = \{\mathcal{C}_1^\tau, \ldots, \mathcal{C}_m^\tau\}$ the set of all sets of reachable connected simple cycles in G^τ. To solve a *UCG* game, we should go over all the (finitely many) possible combinations of $\mathcal{C}_1, \ldots, \mathcal{C}_n$, such that for every $i \in \{1, \ldots, n\}$ we have $\mathcal{C}_i \in \mathcal{C}^{\tau_i}$; and check whether there are vectors $v_1, \ldots, v_n$ such that (i) $v_i \in \mathit{relRatSPAN}(\mathcal{C}_i)$ and (ii) $v_1, \ldots, v_n$ are unsatisfactory with respect to the expression and the threshold; and if such a combination and vector exist, then by Lemma 7, player 1 is the winner.

Hence, in order to solve *UCG* games, we should solve the next problem:

Problem 1 – *Input: a graph (G, s_0), an expression $E = \max(E_1, \ldots, E_m)$ (where E_i is a max-free expression), a threshold ν, and n sets of reachable connected simple cycles $\mathcal{C}_1, \ldots, \mathcal{C}_n$.*

– *Task: Decide if there are vectors $v_1, \ldots, v_n$, such that (i) for every E_i, the max-free constraints , when the average weights of the simple cycles are $v_1, \ldots, v_n$, are infeasible with respect to threshold ν; and (ii) $v_i \in \mathit{relRatSPAN}(\mathcal{C}_i)$ for every $i \in \{1, \ldots, n\}$.*

We note that we can formalize Problem 1 by a first-order formula over the domain of Reals and the dictionary $\langle \mathit{IsRational}(\cdot), +, \times \rangle$, where the predicate $\mathit{IsRational}(\cdot)$ consists of all rational numbers. The need for $\mathit{IsRational}(\cdot)$ arise from the requirement that $v_i \in \mathit{relRatSPAN}(\mathcal{C}_i)$; to overcome this need, we consider a *relaxed version of Problem 1* in which we require $v_i \in \mathit{relSPAN}(\mathcal{C}_i)$ instead of $v_i \in \mathit{relRatSPAN}(\mathcal{C}_i)$.

We denote by MT the greatest lower bound for the threshold ν for which the answer to Problem 1 is YES, and we denote by RMT the corresponding value for the relaxed version of Problem 1. The next lemma states that both values are equal.

Lemma 8 *$MT = RMT$.*

Proof. Clearly, $MT \leq RMT$. In order to prove that $MT \geq RMT$, we define for every max-free expression E_i a function $f_i : \mathbb{R}^n \to \mathbb{R}$, such that $f_i(v_1, v_2, \ldots, v_n) = r$ if r is the maximal threshold for which cycles with average weights $v_1, \ldots, v_n$ are satisfactory for E_i and r. It was established in [1] that f_i is a continuous function, and thus we get that

$$\inf_{v_1 \in \mathit{relSPAN}(\mathcal{C}_1), \ldots, v_n \in \mathit{relSPAN}(\mathcal{C}_n)} f_i(v_1, \ldots, v_n) = \inf_{v_1 \in \mathit{relRatSPAN}(\mathcal{C}_1), \ldots, v_n \in \mathit{relRatSPAN}(\mathcal{C}_n)} f_i(v_1, \ldots, v_n)$$

and the proof of the lemma is completed. □

Since we can encode the relaxed version of Problem 1 by a first-order logic over Reals and $\langle +, \times \rangle$ (also when ν is a variable), then by Tarski's Theorem [4], the value of RMT is computable and hence, by Lemma 7, we get the main result of this section.

Proposition 2 *There exists an effective algorithm to compute the greatest lower bound on ν for which player 1 is the winner in the UCG game (when E and G are given).*

6 The Synthesis Problem for Mean-Payoff Expressions

In this section we consider a game (G, s_0, E, ν), where G is a multidimensional game graph, s_0 is an initial vertex, E is a mean-payoff expression and ν is a threshold. The objective of player 1 is to ensure, by a finite-memory strategy, a value of at most ν for the expression. If such a finite-memory strategy exists, then we say the player 1 is the winner of (G, s_0, E, ν).

The next observation describes the above synthesis problem in terms of unsatisfactory weight vectors.

Observation 1 *Player 1 is the winner of (G, s_0, E, ν) if and only if he a has a finite-memory strategy σ such that the set of simple cycles of any reachable SCC in G^σ is unsatisfactory with respect to E and ν.*

The *winning region* of player 1 is $W_1 = \{s \in S \mid \text{ player 1 is the winner of } (G, s, E, \nu)\}$, that is, all vertices from which player 1 is the winner of the game. In order to solve the problem of whether player 1 is the winner, we compute the winning region W_1, and check whether $s_0 \in W_1$. The next lemma gives a necessary and sufficient condition for the emptiness of the winning region.

Lemma 9 *Player-1 winning region is nonempty if and only if there exists a vertex s such that player 1 is the winner in the UCG game (G, s, E, ν).*

Proof. The direction from right to left is trivial. In order to prove the converse direction, we assume that player-1 winning region is not empty, and we denote by $\sigma = (M, m_0, \alpha_u, \alpha_n)$ the player-1 finite-memory winning strategy for the mean-payoff expression game (G, q, E, ν) (for some $q \in W_1$). We partition G^σ into strongly connected components, and observe that there is at least one end component $\mathcal{EC}$. Let $s^\sigma = (s, m)$ be a vertex in $\mathcal{EC}$. Since the set of reachable cycles in $\mathcal{EC}$ (which is exactly the set of cycles in $\mathcal{EC}$) is unsatisfactory (otherwise, σ is not a winning strategy), we get that $\sigma' = (M, m, \alpha_u, \alpha_n)$ is a winning strategy for the *UCG* game (G, s, E, ν), and the proof follows. $\square$

Analogously to notion of winning regions, we define the notion of *value region.* The value region of a value r in a game is the set of all vertices s for which $\min_{G,s,E} \leq r$. By Lemma 9 we get that the value region of a fixed r, for the mean-payoff expression game, is empty if and only if there exists a vertex s for which $\min_{G,s,E} \leq r$ in the corresponding *UCG* game. Hence, by Proposition 2 we obtain the next recursive algorithm for computing player-1 value regions:

- If the input is an empty graph, then the value regions of all values are empty.
- Compute $\min_{G,s,E}$ in the *UCG* game, for every $s \in S$, and let q be the vertex with the minimal such value, and let r be that value (the computation is possible by Proposition 2). Then $Attr_1(q)$ is part of the value region of r (and any higher values) in the mean-payoff expression game, and we continue the computation, recursively, over the game graph $G - Attr_1(q)$.

Hence, the main result of this paper follows.

Theorem 1 *There is an effective algorithm for the quantitative analysis of mean-payoff expression games, when player 1 is restricted to finite-memory strategies.*

References

1. Krishnendu Chatterjee, Laurent Doyen, Herbert Edelsbrunner, Thomas A. Henzinger, Philippe Rannou. Mean-Payoff Automaton Expressions, In Proc. of CONCUR 2010.
2. Eryk Kopczynski. Half-Positional Determinacy of Infinite Games. In Proc. of ICALP 2006.
3. Yaron Velner. The Complexity of Mean-Payoff Automaton Expression. In Proc. of ICALP 2012.
4. Tarski, A. A Decision Method for Elementary Algebra and Geometry. Manuscript. Santa Monica, CA: RAND Corp., 1948. Republished as A Decision Method for Elementary Algebra and Geometry, 2nd ed. Berkeley, CA: University of California Press, 1951.